# 3. Geometrische Transformationen

Unter geometrischen Transformationen versteht man das Verschieben, Vergrößern und Verkleinern, Drehen, Spiegeln usw. von Objekten innerhalb eines Koordinatensystems oder zwischen Koordinatensystemen. Wir werden die notwendigen Regeln immer für Punkte beschreiben, dadurch lassen sich alle anderen Objekte auch leicht transformieren indem deren definierende Punkte transformiert werden.

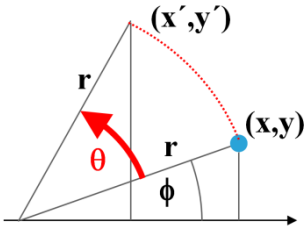
## Einfache 2D-Transformationen

### Translation (Verschiebung)

Das Verschieben eines Punktes (x,y) um den Vektor $(t_x,t_y)$ liefert den transformierten Punkt

$$(x',y') = (x+t_x,y+t_y) \quad .$$

### Rotation (Drehung)




Durch das Drehen eines Objektes mit dem Winkel θ um den Koordinatenursprung kommt der Punkt (x,y) wegen $x = r\cdot\cos\Phi$ und $y = r\cdot\sin\Phi$ →

$x' = r\cdot\cos(\Phi + \theta) = r\cdot\cos\Phi\cdot\cos\theta - r\cdot\sin\Phi\cdot\sin\theta = x\cdot\cos\theta - y\cdot\sin\theta$ (und y' analog) auf

$$(x',y') = (x\cdot\cos\theta - y\cdot\sin\theta,\ x\cdot\sin\theta + y\cdot\cos\theta)$$

zu liegen.

### Skalierung (Vergrößerung oder Verkleinerung)

Beim Skalieren eines Objektes um den Faktor s um den Ursprung (0,0) wird ein Punkt (x,y) auf

$$(x',y') = (s\cdot x,\ s\cdot y)$$

abgebildet.

Wenn in x- und y-Richtung unterschiedliche Skalierungsfaktoren $s_x$ und $s_y$ verwendet werden, dann erhält man

$$(x',y') = (s_x\cdot x,\ s_y\cdot y) \quad .$$

### Reflexion (Spiegelung)

Die Spiegelung an einer Koordinatenachse ist ein Sonderfall der Skalierung mit $s_x = -1$ oder $s_y = -1$.

Alle anderen Transformationen können durch Hintereinanderausführen der beschriebenen einfachen Transformationen erreicht werden. Diese Abbildungen (mit Ausnahme der Translation) lassen sich auch durch ***Transformationsmatrizen*** darstellen. Dabei werden die Punkte als Vektoren dargestellt, um mit ihnen die Matrixoperationen durchführen zu können:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Skalierung

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation (gegen Uhrzeigersinn)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

Spiegelung um x-Achse

$$(x'\ y') = (x+t_x,\ y+t_y)$$ ... ?

Verschiebung

## Homogene Koordinaten

Damit auch die Translation in Matrixschreibweise angegeben werden kann, verwendet man *homogene Koordinaten*. Jedem Punkt wird eine zusätzliche Koordinate h zugeordnet, wobei die Umrechnung in 2D-Koordinaten durch Division der x- und y-Komponente durch h erfolgt. Daher verwendet man meist h=1. Für den Punkt (x,y) schreiben wir daher (x,y,1). Die Transformationsmatrizen werden um eine Zeile und Spalte mit Einheitswerten erweitert:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Warum ist es vorteilhaft, alle Transformationen in einheitlicher Matrixschreibweise zu formulieren? Meist werden größere Teile (Objekte, Bilder) als Ganzes transformiert, d.h. auf jeden Punkt dieser Gebilde wird die gleiche Folge von Transformationen angewendet. Dies entspricht einer sequenziellen Multiplikation eines Punktes P mit Matrizen $M_1$, $M_2$, $M_3$, … : $P' = M_1 \cdot P$, $P'' = M_2 \cdot P'$, $P''' = M_3 \cdot P''$, … . Nun kann man sich die Assoziativität der Matrixmultiplikation [also $(M_1 \cdot M_2) \cdot M_3 = M_1 \cdot (M_2 \cdot M_3)$ ] zunutze machen und den Rechenaufwand damit massiv reduzieren:

**Statt $P^{(n)} = M_n \cdot (M_{n-1} \cdot \ldots (M_3 \cdot (M_2 \cdot (M_1 \cdot P))) \ldots)$ schreibt man $P^{(n)} = (M_n \cdot M_{n-1} \cdot \ldots \cdot M_3 \cdot M_2 \cdot M_1) \cdot P$ .**

Nun kann man $M = (M_n \cdot M_{n-1} \cdot \ldots \cdot M_3 \cdot M_2 \cdot M_1)$ *vorher* ausrechnen und diese *eine* Gesamtmatrix dann auf alle Punkte anwenden.

Wir wollen den einfachen Transformationsmatrizen zur übersichtlicheren Darstellung Namen geben:

**$T(t_x,t_y)$** = Translation um den Vektor $(t_x,t_y)$
**$R(\theta)$** = Rotation um den Winkel $\theta$ [pos. Winkel = Rotation gegen den Uhrzeigersinn!]
**$S(s_x,s_y)$** = Skalierung um die Faktoren $s_x$ und $s_y$.

Die inversen Transformationen dieser einfachen Transformationen sind:

$T^{-1}(t_x,t_y) = T(-t_x,-t_y)$ $R^{-1}(\theta) = R(-\theta)$ $S^{-1}(s_x, s_y) = S(1/s_x, 1/s_y)$

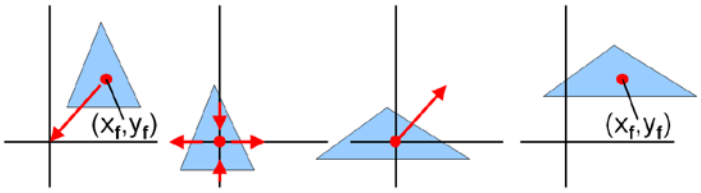
Aus diesen einfachen Transformationen lassen sich nun auch komplexere Transformationen herleiten.
Als Beispiel betrachten wir die

**Skalierung um einen anderen Punkt als den Koordinatenursprung:**

1. Schritt = Verschieben des Skalierungszentrums in den Koordinatenursprung: $T(-x_f,-y_f)$
2. Schritt = Skalieren des Objektes im Koordinatenursprung: $S(s_x,s_y)$
3. Schritt = Zurückverschieben des Objektes an die ursprüngliche Stelle: $T^{-1}(-x_f,-y_f) = T(x_f,y_f)$

Die allgemeine Matrix für die Skalierung mit $(x_f,y_f)$ als Zentrum erhält man nun so:

$S(x_f,y_f,s_x,s_y) = T(x_f,y_f) \cdot S(s_x,s_y) \cdot T(-x_f,-y_f)$




Als weiteres Beispiel betrachten wir die

**Spiegelung an einer beliebigen Achse y = mx+b:**

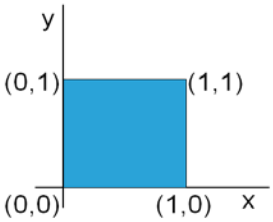
1. Schritt = Verschieben, so dass die Achse durch den Koordinatenursprung geht: $T(0,-b)$
2. Schritt = Drehen, so dass die Achse z.B. mit der x-Achse zusammenfällt: $R(-\theta)$ [$m = \tan\theta$]
3. Schritt = Spiegeln an der x-Achse: $S(1,-1)$
4. Schritt = Zurückdrehen, so dass Achse ursprünglichen Winkel hat: $R^{-1}(-\theta) = R(\theta)$
5. Schritt = Zurückverschieben, so dass Achse an der ursprüngliche Stelle liegt: $T^{-1}(0,-b) = T(0,b)$

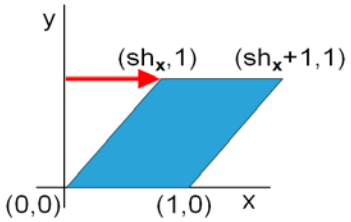
Die allgemeine Matrix für die Spiegelung an der Achse y = mx+b erhält man nun so:

$X(m,b) = T(0,b) \cdot R(\theta) \cdot S(1,-1) \cdot R(-\theta) \cdot T(0,-b)$

Eine weitere wichtige Transformation ist die *Scherung*, sie hat im einfachsten Fall in x-Richtung mit fixierter x-Achse die Form:

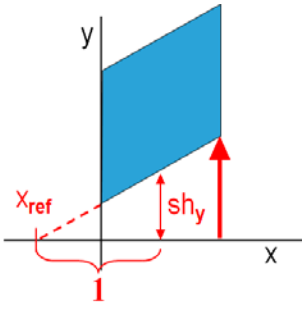
$$\begin{pmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$




Etwas allgemeiner kann die Scherung auch an einer Parallelen zu einer Achse erfolgen, das sieht etwa in y-Richtung dann so aus:

$$\begin{pmatrix} 1 & 0 & 0 \\ sh_y & 1 & -sh_y \cdot x_{ref} \\ 0 & 0 & 1 \end{pmatrix}$$




Natürlich kann man jetzt auch wieder ganz leicht die allgemeine Matrix für eine Scherung ableiten, die nicht parallel zu einer der Koordinatenachsen erfolgt: zuerst Drehung in achsenparallele Lage, dann Scherung, und dann wieder zurückdrehen.

Auch die allgemeine Viewing-Transformation lässt sich ganz leicht mit Transformationsmatrizen beschreiben. Die verwendeten Operationen sind Verschiebung eines Ursprunges in den anderen, Drehung des Viewports in die Achsenrichtungen des Windows und Skalierung der Achsen. Im Unterschied zu den oben behandelten zusammengesetzten Transformationen erfolgt aber hier kein Zurückdrehen, Zurückverschieben etc.

**Affine Transformationen.** Alle hier behandelten Transformationen sind *affine Transformationen*, d.h. die Koordinaten lassen sich durch lineare Funktionen plus einer Translation ineinander überführen. Affine Abbildungen erhalten Kollinearität, d.h. (je) 3 Punkte auf einer geraden Linie sind auch nach der Abbildung auf einer geraden Linie, und die Proportionalität von Abständen entlang einer gerade Linie, d.h. das Verhältnis von Längen auf einer Geraden, bleibt erhalten. Weiters bleiben parallele Linien immer parallel und endliche Punkte bleiben im Endlichen. Alle affinen Transformationen lassen sich aus Skalierung, Rotation und Translation zusammensetzen (auch die Scherung!). Affine Transformationen, bei denen nur Rotation, Translation und Spiegelung verwendet werden, sind überdies längen- und winkelerhaltend.

## 3D Transformationen

Alle 2D-Konzepte lassen sich leicht auf 3D erweitern. Man benötigt wieder eine homogene Komponente, so dass 4x4-Matrizen auf 4-dimensionalen Vektoren operieren. Später werden wir sehen, dass man auch Projektionen auf diese Art formulieren kann.

Hier sind einmal die wichtigsten 3D-Transformationen:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D-Translation | 3D-Skalierung | Spiegelung um yz- | xz- | xy-Ebene

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

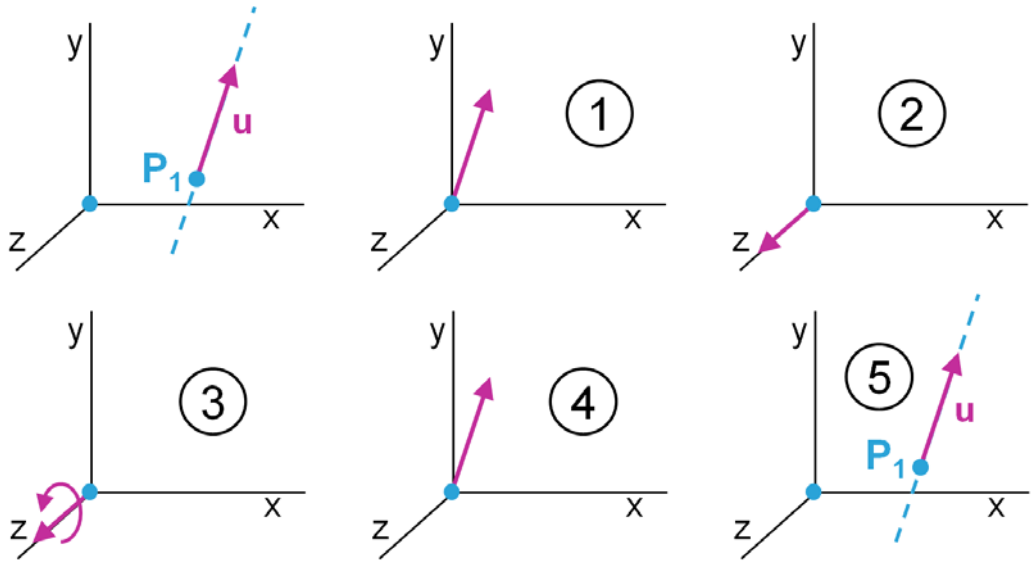
3D-Rotation um x-Achse | 3D-Rotation um y-Achse | 3D-Rotation um z-Achse

Die Namen für diese einfachen Transformationsmatrizen sehen nun so aus:

**$T(t_x,t_y,t_z)$** = Translation um den Vektor $(t_x,t_y,t_z)$
**$R_x(\theta)$** = Rotation um den Winkel θ um die x-Achse; y- und z-Achse analog
**$S(s_x,s_y,s_z)$** = Skalierung um die Faktoren $s_x$, $s_y$ und $s_z$.

Als Beispiel für eine komplexere Transformation wollen wir eine

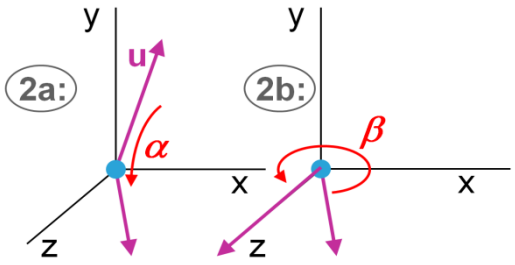
**Drehung um den Winkel θ um eine beliebige Achse im Raum**
herleiten. Die Achse sei durch einen Punkt $P_1(x_1,y_1,z_1)$ und einen Richtungsvektor u gegeben.



1. Schritt = Punkt $P_1$ in den Koordinatenursprung verschieben: $T(-x_1,-y_1,-z_1)$
2. Schritt = Vektor u in die z-Achse drehen
   2a. Vektor u um die x-Achse in die xz-Ebene drehen: $R_x(\alpha)$
      Sei u = (a,b,c), dann ist u'=(0,b,c) die Projektion von u auf die yz-Ebene. Der Drehungswinkel α um die x-Achse ergibt sich aus cos α = c/d mit d= $\sqrt{(b^2+c^2)}$
   2b. Vektor u um die y-Achse in die z-Achse drehen: $R_y(\beta)$
      Der Drehungswinkel β um die y-Achse ergibt sich aus cos β = d (bzw. sin β = -a)
3. Schritt = Drehung um θ um die z-Achse ausführen: $R_z(\theta)$
4. Schritt = Vektor u in die ursprüngliche Richtung zurückdrehen: zuerst $R_y(-\beta)$, dann $R_x(-\alpha)$
5. Schritt = Punkt $P_1$ an die ursprüngliche Position zurückverschieben: $T(x_1,y_1,z_1)$

Die resultierende Matrix berechnet sich also so:

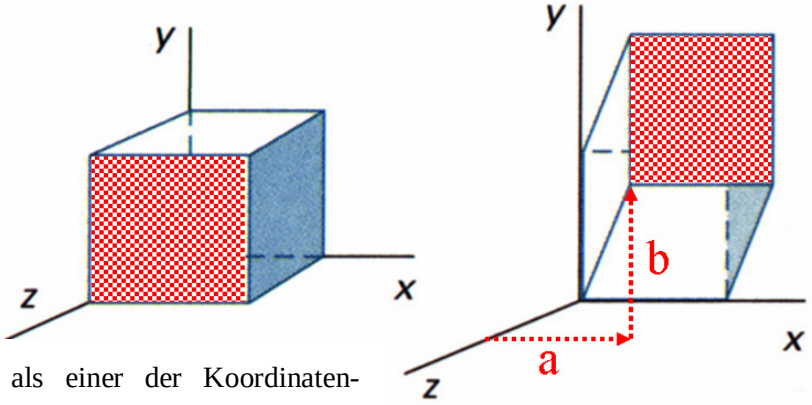
$$R(\theta) = T^{-1}(-x_1,-y_1,-z_1)\cdot R_x^{-1}(\alpha)\cdot R_y^{-1}(\beta)\cdot R_z(\theta)\cdot R_y(\beta)\cdot R_x(\alpha)\cdot T(-x_1,-y_1,-z_1) =$$
$$= T(x_1,y_1,z_1)\cdot R_x(-\alpha)\cdot R_y(-\beta)\cdot R_z(\theta)\cdot R_y(\beta)\cdot R_x(\alpha)\cdot T(-x_1,-y_1,-z_1)$$




Die ***Scherung in 3D*** ist ebenfalls einfach darstellbar:

Eine Scherung parallel zur xy-Ebene um den Wert a in x-Richtung und den Wert b in y-Richtung erreicht man mittels

$$\begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$




Eine Scherung mit einer anderen Fixebene als einer der Koordinatenhauptebenen kann man sich leicht ableiten.